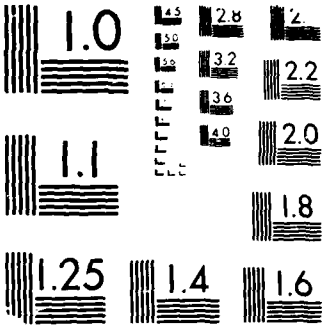
AD-A191 634 GENERALIZATION AND THE BACKWARD PROPAGATION NEURAL NETWORK(U) BROWN UNIV PROVIDENCE R I DEPT OF PHYSICS C M BACHMANN 14 JAN 88 ARO-22000.9-LS DAAG29-84-K-0202 1/1

UNCLASSIFIED F/G 21/3 NL



MICROCOPY RESOLUTION TEST CHART

(2)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
ARO 22000.9-LS	N/A	N/A

TITLE (and Subtitle)

Generalization and the Backward Propogation Neural Network

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER

AUTHOR(s)

Charles M. Bachmann

8. CONTRACT OR GRANT NUMBER(s)

DAAG-29-84-K-0202

PERFORMING ORGANIZATION NAME AND ADDRESS

Department of Physics and Center for Neural Science, Brown University, Providence, RI 02912

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

N/A

11. CONTROLLING OFFICE NAME AND ADDRESS

U. S. Army Research Office
Post Office Box 12211
Research Triangle Park, NC 27709

12. REPORT DATE

January 14, 1988

13. NUMBER OF PAGES

10 pages

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)

Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

NA

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Backward Propogation, Generalization

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Results of simulations in discrete and continuous input simulations are discussed for Rumelhart's Backward Propogation of Error Neural Network. Comparison of the results offers a way to understand the problem of generalization from a partial set of data in the Backward Propogation Network.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Generalization and the Backward Propagation Neural Network

Charles M. Bachmann

December 19, 1987

1 Introduction

The capacity of model neural networks to generalize from a partial set of information is an area of much current interest. In some sense it addresses the very issue of how accurate current models are of higher cognitive processes, for the ability to categorize input, to make generalizations based on a limited set of information, is one of the hallmarks of these processes.

In this context, ~~we have~~ been investigating the Backward Propagation of Error Model due to Rumelhart et. al.[1]. The model is a deterministic approach which seeks to teach a desired input-output mapping by repeated presentation of the desired mapping to the system, correcting the system connections based on the error in output. We have begun to address the generalization capability of this system. Specifically, we have studied to what extent the set of connections which evolve in learning a partial set of patterns are a general solution to a given mapping. That is, if we teach several examples of a mapping to the system, will the solutions that the system discovers for these patterns be capable of generalizing and correctly identifying other input states that have not been seen. The results of some simulations undertaken to address this question are discussed and some modifications to the model which we have proposed are indicated.

2 Rumelhart's Backward Propagation

Rumelhart's model[2] consists of an input and output layer of neurons with one or several layers of neurons, "hidden units", in between. A set of input states, $\vec{f}^s$, s = 1,...,p, is presented repeatedly to the system until the system reaches an output for each input pattern which is close to the target. [1] The output of the system, $\vec{o}^s$, is compared with the target $\vec{\tau}^s$, and the weights between the output layer and the last hidden layer are modified. The error signal is then propagated back accross these modified weights to the next layer of connections, where the weights are modified. In this way, the error signal at the output layer is passed backward layer by layer to modify all of the connections systematically. A simple example with one "hidden layer" of neurons is shown in figure 1. For figure 1, the equations summarizing

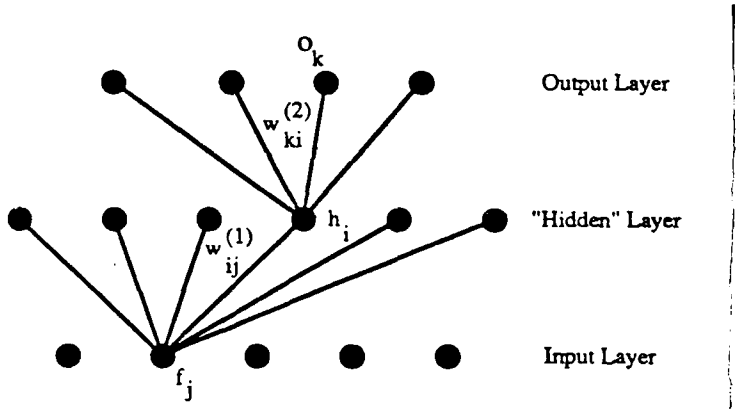


Figure 1: The o_k label the output units, the h_i label the "hidden units", and the f_j label the input units. Only some of the connections are shown. Superscripts on synaptic weights denote layer index.

the forward propagation of an input signal $\vec{f}^s$ (presentation of the training pattern $\vec{f}^s$) are :

$$x_i^s = \sum_{j=1}^{N_1} w_{ij}^{(1)} f_j^s + \phi_i^{(1)} \qquad (1)$$

[1] Since the targets are usually binary-valued and can only be reached asymptotically, a distance from target learning cutoff must be specified. We typically chose 0.1

$$h_i^s = \psi(x_i^s) \tag{2}$$

$$y_k^s = \sum_{i=1}^{N_2} w_{ki}^{(2)} h_i^s + \phi_k^{(2)} \tag{3}$$

$$o_k^s = \psi(y_k^s) \tag{4}$$

where $\psi(z)$ is the sigmoid input-output function given by: [2,3]

$$\psi(z) = \frac{1}{1+e^{-\lambda z}}. \tag{5}$$

In effect, to learn the correct response, the system must minimize the difference between the target output, $\vec{\tau}^s$, and the output $\vec{o}^s$ for each pattern. The modifications to the connections, therefore, are made as a gradient descent in the energy function:

$$\xi = \frac{1}{2}\sum_{s=1}^{p}\sum_{k=1}^{N_2}(o_k^s - \tau_k^s)^2. \tag{6}$$

To simplify computation modifications are made after each pattern is presented, so that as an approximation to true gradient descent, the gradient is computed actually from the partial energy :

$$\xi_s = \frac{1}{2}\sum_{k=1}^{N_2}(o_k^s - \tau_k^s)^2. \tag{7}$$

In the above example, the modification to the weights $w_{ki}^{(2)}$ after the presentation of pattern s is given by:

$$\begin{aligned}\Delta_s(w_{ki}^{(2)}) &= -\alpha\frac{\partial \xi_s}{\partial w_{ki}^{(2)}} \\ &= -\alpha\frac{\partial \xi_s}{\partial o_k^s}\frac{\partial o_k^s}{\partial y_k^s}\frac{\partial y_k^s}{\partial w_{ki}^{(2)}} \\ &= -\alpha\lambda(o_k^s - \tau_k^s)o_k^s(1-o_k^s)h_i^s,\end{aligned} \tag{8}$$

[2] In Rumelhart's original model there is no gain parameter λ, but we have found it useful to introduce this parameter to study the effect of varying the strength of the nonlinearity by using it to change the size of the region over which the function is in fact approximately linear, which is $\sim O(\frac{1}{\lambda})$.

[3] Hopfield also used a gain parameter in the continuous version of his model [1] to study the effect of changing the character of the nonlinearity.

where we have used the fact that the derivative of the sigmoid input-output function in equation 5 is $\psi'(z) = \lambda\psi(z)(1-\psi(z))$; α is a small positive constant. Similarly, after the weights between the last layer and the hidden layer have been updated, the error signal is propagated back across these new weights, and the weights between the input layer and the hidden layer are then updated, according to:

$$\begin{aligned}\Delta_s(w_{ij}^{(1)}) &= -\alpha\frac{\partial\xi_s}{\partial w_{ij}^{(1)}} \\ &= -\alpha\sum_{k=1}^{N_2}\frac{\partial\xi_s}{\partial o_k^s}\frac{\partial o_k^s}{\partial y_k^s}\frac{\partial y_k^s}{\partial h_i^s}\frac{\partial h_i^s}{\partial x_i^s}\frac{\partial x_i^s}{\partial w_{ij}^{(1)}} \\ &= -\alpha\lambda^2\sum_{k=1}^{N_2}(o_k^s-\tau_k^s)o_k^s(1-o_k^s)w_{ki}^{(2)}h_i^s(1-h_i^s)f_j^s. \qquad (9)\end{aligned}$$

In the model biases are treated as being the connection strength from units which always have the output value 1; as a consequence, they are also modified when the error signal is propagated backward. One additional feature of the modification procedure is a "momentum" term which is added at each step of the modification procedure to diminish the effect of oscillation in the gradient descent search for the minimum; the "momentum" term consists of adding a small amount proportional to the previous modiciation, so that the actual modification at time step t for the nth layer of synaptic connections is:

$$\Delta_{s(t)}(w_{ki}^{(n)}) = -\alpha\frac{\partial\xi_{s(t)}}{\partial w_{ki}^{(n)}} + \kappa\Delta_{s(t-1)}(w_{ki}^{(n)}) \qquad (10)$$

where s(t) denotes the index of the pattern presented at time step t and κ is a small positive constant less than 1. One can also write this in a slightly different form:

$$\Delta_{s(t)}(w_{ki}^{(n)}) = -\eta\sum_{\nu=0}^{t}\frac{\partial\xi_{s(\nu)}}{\partial w_{ki}^{(n)}}\kappa^{t-\nu}. \qquad (11)$$

This form is more suggestive in showing that the use of a momentum term is equivalent to a discrete approximation to a "temporal" integral average with an exponentially decaying kernel. The kernel has the effect of weighting the influence of the patterns most recently presented, assigning exponentially less weight to those patterns presented earlier in time.

3 Simulations and Results

In simulations of the Backward Propagation of Error Model, we have studied the ability of the model to generalize based on a limited training set. We have studied several paradigms thus far. In the realm of discrete binary-valued pattern identification, we have considered two problems: (1)the identification of the parity of the input vector with four and eight input units and (2)the boolean mapping AND on two input units in a four-dimensional input space. At the same time we have also begun to consider pattern recognition of patterns in continuous spaces, specifically the identification and separation of geometrical shapes and regions which are not necessarily simply connected. Contrasting the results of the discrete pattern spaces with those of the continuous space problems offers some insight into the generalization properties of the Backward Propagation network.

The problem of the AND mapping for binary input consists of training the network to respond with a one if two particular bits in the input pattern are both on but to respond with a zero otherwise. This problem then is equivalent to identification of a subspace; for example, for a four-dimensional input vector, the network is asked to isolate the subspace consisting of one particlur two-dimensional lattice from the rest of the space. For the AND problem, the decision space, therefore, is not terribly complex (a single hyperplane separates the two subspaces), and the Backward Propagation network is capable of some generalization.

In contrast, for the parity problem, where the pattern class boundaries are more complex and are, in fact, disjoint, the system is incapable of generalization. Specifically, the parity problem consists of determining whether a binary input vector has an an even or an odd number of ones. For an odd number of ones, the correct response is one, and for an even number of ones, the correct response is zero. The problem has a solution provided that there are a sufficient number of hidden units ($\geq N$) [1] and provided that the entire training set of 2^N patterns is specified. We have found, however, that when the neural network is trained on a subset of the 2^N patterns, it will usually not correctly identify patterns which were not presented to it during the training phase, regardless of what percentage of the possible patterns have previously been shown. In fact, the network tends to respond as if the previously unseen pattern had a parity dichotomous to

its actual parity. Comparing this result with that obtained for the AND problem suggests that the disjoint decision boundaries make it difficult for the system to arrive at a truly "general" solution. However, comparing this outcome with a paradigm for a continuous space problem, which we will presently describe, can help make the distinction more precise.

In the realm of continuous space generalization, we have begun to study a paradigm which consists of teaching the network a geometric to distinguish different regions, not all of which are simply connected, in a continuous space. The value of each input neuron corresponds to one of the cartesian coordinates of a given point in the space. The inputs,therefore, can take on both positive and negative real values. As an example of this, we have considered a problem, in which concentric circles define two regions, an inner disk and an outer annulus. The mapping to be learned consists of identifying any point(pattern) in the inner disk with an output of one and any point(pattern) in the outer annulus with an output of zero. Training patterns are selected randomly and uniformly throughout the two regions; the x- and y- coordinates are the values of the two input neurons respectively for each pattern. After training we find that when we select new unseen patterns at random, the system will generalize what it has learned and correctly identify a large fraction of the previously unseen inputs. In the following figure, we show a generalization curve for a simulation which had four hidden units and which was trained with 150 randomly chosen patterns. The concentric circles had inner and outer radii of 1 and 2 respectively. The generalization curve plots output versus radius at incremental radii of (1/15) for patterns not seen during the training phase. At each radius, 100 patterns are chosen at random angles, the x and y coordinates of each point are presented to the network, and an output is determined; the generalization curve plots the mean and standard deviation of the 100 patterns at each radius. Provided a sufficient number of patterns are chosen and also provided that we do not saturate the system with too many training patterns, the network develops a generalization curve approaching step-function which would occur if the network were performing the ideal mapping.

If we compare the concentric circle problem with the two discrete-input paradigms, we may infer some of the limitations of the limitations of the generalization capabilities of the current Backward Propagation Model. It

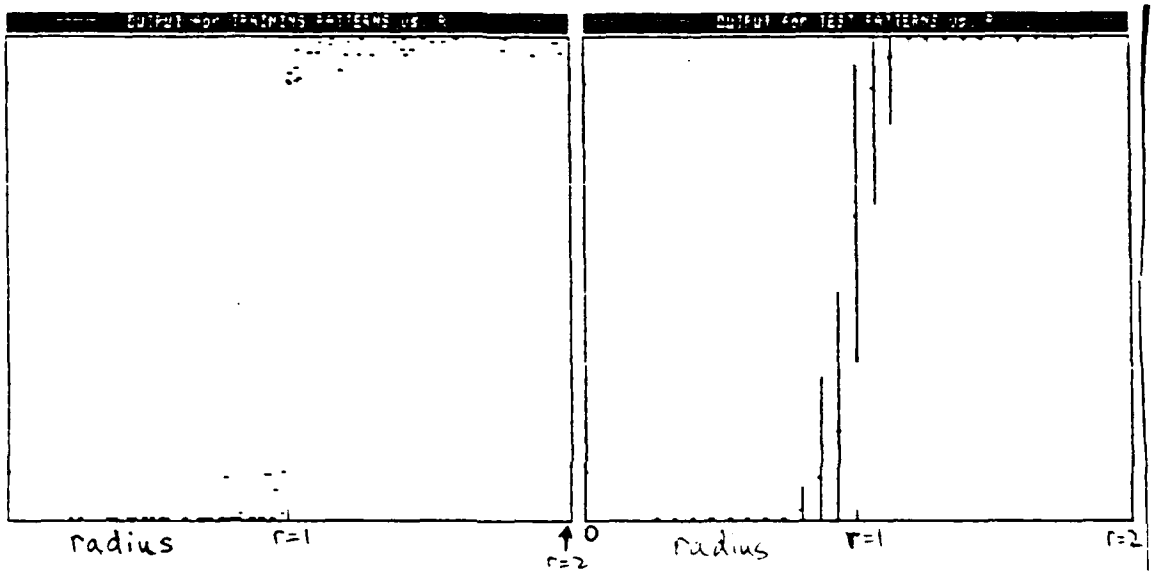


Figure 2: (Left)Output of training patterns as a function of radius. (Right)Generalization Curve: output of patterns not seen during training period as a function of radius.

is logical that the problem in the continuous space with continuously connected regions should be easier for the network to learn about. Except near the boundary, any point and another point sufficiently close to it will have the same output. Thus, provided that enough random points are selected in the two regions, the system can learn to identify unseen points from what it has seen previously because there are training patterns that have been learned which will be sufficiently close to it with the same output. Indeed in the example shown, the only region of ambiguity and large standard deviation is near the boundary. In contrast, in the discrete lattice defined by the parity problem, for any point on the lattice with a given parity, all of its nearest neighbors will have the opposite parity. This leads to problems in generalization, then, because the level of analysis on which the system appears to discriminate is to assign an output for an unseen pattern equal to that of training pattern(s) which are closest; in the parity problem, this will very often be a nearest-neighbor pattern, which will have the opposite parity. In the simulations which we ran for the parity problem with a four-dimensional input space, we ran simulations in which we presented 12 of the 16 possible binary inputs, which meant that for each of the four patterns not presented until after the learning phase, there was at least one nearest neighbor, which of course had opposite parity, and which had been used in training. The AND problem, on the other hand, is similar to the concentric circle problem in that neighboring states through much of the lattice will have the same output; also, pattern class boundaries are not disjoint, and only near the boundary can there be potential confusion.

4 Conclusions and Future Directions

If the Backward Propagation Model is to be useful for problems like that of parity, it must be improved to the point where it analyzes incoming data in a more sophisticated manner than its current formulation. It is not surprising that it has difficulty learning a problem like parity. The learning procedure considers only one pattern at a time, with only a weak memory of previous modifications being provided by the "momentum" term. Essentially, it treats the gradient descent by descending down the individual pattern troughs instead of a global trough. If we only care about learning

the training set in a problem like parity, this will be sufficient usually; however, if we wish the network to generalize, it must have a better means of comparing and/or storing information about more than pattern.

One approach that we hope to try, therefore, is a modified energy functional and learning procedure. We propose a procedure in which patterns are optimized in random pairs and the energy functional is either a product of the square errors of each of the two patterns,

$$\xi = (o^{s_1} - \tau^{s_1})^2 \cdot (o^{s_2} - \tau^{s_2})^2, \tag{12}$$

or else a linear sum of the two energies:

$$\xi = (o^{s_1} - \tau^{s_1})^2 + (o^{s_2} - \tau^{s_2})^2. \tag{13}$$

In the latter case, the principal difference between the suggested implementation and the original algorithm is that of ordering. But this may be an important point, for consider the following. In the binary learning paradigms in which the value of inputs take on only the value of 1 or 0, the sign of the modification of a given weight is entirely determined by the difference $(\tau^s - o^s)$. This means in effect that patterns in different classes are always trying to modify the weights in the opposite direction. In the original algorithm, the patterns are always presented in the same order, which means that in effect patterns which are very far apart on the list are never really compared. By choosing pairs at random (perhaps one from each class), the system may be able to "compare" many different pairs of data and draw better class distinctions. The product energy functional may also have some additional merits, but this is unclear at this time.

An alternative approach may also be to add additional symmetry searching terms to the energy functional.[4] However, an appropriate form for such terms has yet to be constructed.

[4] An approach suggested by Amir Dembo and Ofer Zeitouni, personal communication

END

DATE

FILMED

5-88

DTIC